P^{3}Forecast: Personalized Privacy-Preserving Cloud Workload Prediction based on Federated Generative Adversarial Networks

Yu Kuang*, Li Yan*, Zhuozhao Li†
*School of Cyber Science and Engineering, Xi'an Jiaotong University, China
†Department of Computer Science and Engineering, Southern University of Science and Technology, China
Email: yooki.k@stu.xjtu.edu.cn, li.yan.88@xjtu.edu.cn, lizz@sustech.edu.cn

***Abstract*—To comply with privacy regulations and self-protection from cloud outages, increasingly more users are leveraging multiple cloud providers for service deployment. However, due to the isolation of highly Non-Identically and Independently Distributed (Non-IID) workload datasets and deficiencies of existing methods as reflected in our experimental studies, no cloud providers can single-handedly capture the workload patterns of such users for accurate workload prediction. Accordingly, we propose *P^{3}Forecast*, a Personalized Privacy-Preserving Cloud workload prediction framework based on Federated Generative Adversarial Networks (GANs), which allows cloud providers with Non-IID workload data to collaboratively train workload prediction models as preferred while protecting privacy. We first design a data synthesis quality assessment method based on Dynamic Time Warping (called *pattern-aware DTW*), which is insusceptible to time series length and reliable for the comparison of temporal patterns. By using *pattern-aware DTW* as the model aggregation weights, we adopt the Federated Learning (FL) of a GAN model for the augmentation of IID workload training datasets per cloud provider. Then, we further design a post-training method of local workload prediction models, which consists of a query mechanism for extracting the most informative synthesized data for training dataset augmentation and a learning rate adjustment strategy for stable convergence. Extensive experiments driven by real-world workloads demonstrate that compared with the state-of-the-art, *P^{3}Forecast* improves workload prediction accuracy by 19.5%-46.7% in average over all cloud providers, while ensuring the fastest convergence in Federated GAN training.**

***Index Terms*—Cloud Workload Prediction, Federated Learning, Generative Adversarial Networks**

I. Introduction

With the popularity of new cloud applications (e.g., Machine-Learning-as-a-Service), workloads are becoming increasingly heterogeneous and dynamic, which hinders the accurate prediction of cloud workloads and predictive orchestration of cloud resources [1], [2].

Multiple traditional approaches have been developed for workload prediction of individual cloud providers [1], [3]–[6]. Generally, they utilize various Machine Learning (ML) techniques, such as Autoregressive Integrated Moving Average (ARIMA), Deep Neural Networks (DNN), to capture the complex patterns and high-dimensional features of workloads. However, as cloud providers aided by traditional workload prediction paradigms often rely on isolated workload databases, they may struggle to adapt to varying workload types. For example, public cloud providers, such as *Alibaba*, have extensive experience in handling ML workloads, but lack sufficient data to learn patterns associated with scientific computing workloads [2]. Under such cases, traditional approaches are unable to versatilely predict workloads covering various service categories, which makes it challenging for users to leverage the workload prediction results for optimal deployment of their applications across multiple cloud platforms [7].

Nevertheless, it is neither practical nor secure to share workload traces across providers due to the high expense of network transmission and risk of data privacy leakage [8], [9]. Federated Learning (FL) [10], which enables collaborative ML model training without exposing participants' privacy-sensitive data, shines a light on establishing a versatile workload prediction framework via collaboration across providers [7], [11]. Despite this, our preliminary studies (Section III-D) show that the workloads of different cloud providers are often highly Non-Identically and Independently Distributed (Non-IID) due to their different cloud computing environments and diverse business involvements, and may cause significant performance drop in FL training [12], [13].

In recent years, numerous approaches have been proposed to tackle Non-IID issues in FL [13]–[18]. Among them, the methods leveraging shared Generative Adversarial Networks (GANs) to augment FL training data, known as Virtual Homogeneity Learning (VHL), [15], [16], [19] stand out for faster convergence and stronger generalization performance. However, these methods mostly focus on tasks involving spatial data (e.g., image classification task), and are therefore unsuitable for cloud workload prediction, which relies on time series data with abundant temporal features. Moreover, to augment the FL training data, these methods require participants to share the base noise that covers all heterogeneous workload features (e.g., CPU usage, requested memory amount) across cloud providers, which is against the privacy regulations of most cloud providers [7].

A possible solution is to train the GAN model in a FL manner [20]–[25]. Thus, each client can use the trained GAN model to synthesize IID training data. However, different from the model aggregation strategy in classic FL tasks, of which model updates are generally averaged or weight-averaged by data quantity or data distribution discrepancy , the contribution of clients in the FL training of time series GAN models should be scored by the quality of synthesized time series data for faster convergence and better data synthesis capability [23], [24]. Although Dynamic Time Warping (DTW) has been widely utilized to assess temporal data synthesis quality in GAN-based data augmentation [26], [27], our experimental studies in Sections III-E and III-F demonstrate that DTW is

unreliable for workload data synthesis quality assessment, and existing methods are inefficient due to the lack of personalization in data augmentation.

Considering that temporal features are vital for effective workload data augmentation under Non-IID settings, while existing methods suffer from unreliability in data synthesis quality assessment and inefficiency in data augmentation, we propose *P³Forecast*, a **P**ersonalized **P**rivacy-**P**reserving Cloud Workload Prediction framework based on Federated GAN, which allows cloud providers with heterogeneous workload data to collaboratively train workload prediction models according to their preference while protecting privacy. Specifically, we first incorporate the measurement of the first-order difference between time series and normalization into DTW to ensure that our proposed data synthesis quality assessment method (called the *pattern-aware DTW*) is insusceptible to time series length and reliable for the comparison of temporal patterns. Then, by using the *pattern-aware DTW* as the weight of model updates from the cloud providers, we adopt FL for the collaborative training of a GAN model, which can be utilized by each cloud provider to synthesize its own IID training data. Finally, we design a post-training method of local workload prediction models, which consists of a query mechanism for extracting the most informative synthesized data for training dataset augmentation and a learning rate adjustment strategy for stable convergence. Unlike most existing FL personalization methods that require consistent model architectures across clients, *P³Forecast* allows each cloud provider to specify its workload prediction model architecture as preferred, which is especially suitable for cloud providers with heterogeneous model preferences and data augmentation preferences. The contributions are summarized as follows:

(1) We conduct extensive experimental studies on large-scale workload datasets collected from seven cloud clusters or centers to demonstrate the Non-IID nature of cloud workloads and deficiencies of existing methods in realizing the collaborative training of workload prediction models.
(2) We propose *P³Forecast*, a personalized privacy-preserving cloud workload prediction framework based on Federated GAN. It utilizes a novel DTW-based data synthesis quality assessment method for the aggregation of Federated GAN models, which is both insusceptible to sequence length and reliable for temporal pattern comparison, and a post-training method of workload prediction models, which employs a query mechanism and a learning rate adjustment strategy for personalized data augmentation and stable model training.
(3) We conduct detailed experiments on real-world cloud workloads. Compared with the state-of-the-art, *P³Forecast* drastically improves workload prediction accuracy by 19.5%-46.7% in average over all cloud providers, while ensuring the fastest convergence in Federated GAN training.

The rest of this paper is organized as follows. Section II summarizes related works. Section III presents preliminaries and motivations. Section IV introduces the detailed design of *P³Forecast*. Section V presents performance evaluation results. Section VI concludes the paper with future remarks.

II. Related Work

Traditional Cloud Workload Prediction. Early efforts generally utilize regression-based approaches for cloud workload prediction. For example, Calheiros *et al.* [3] proposed an ARIMA-based approach to predict future workloads generated by web application requests. Kim *et al.* [1] proposed to ensemble multiple predictors for cloud workload prediction. Recent research mostly applies deep learning models (such as Long Short-Term Memory (LSTM) [4] and Gated Recurrent Units (GRU) [5], [6]) for cloud workload prediction. However, due to limited and isolated data of individual cloud providers, these methods cannot establish a versatile workload prediction model capable of accurately predicting workloads covering various service categories.

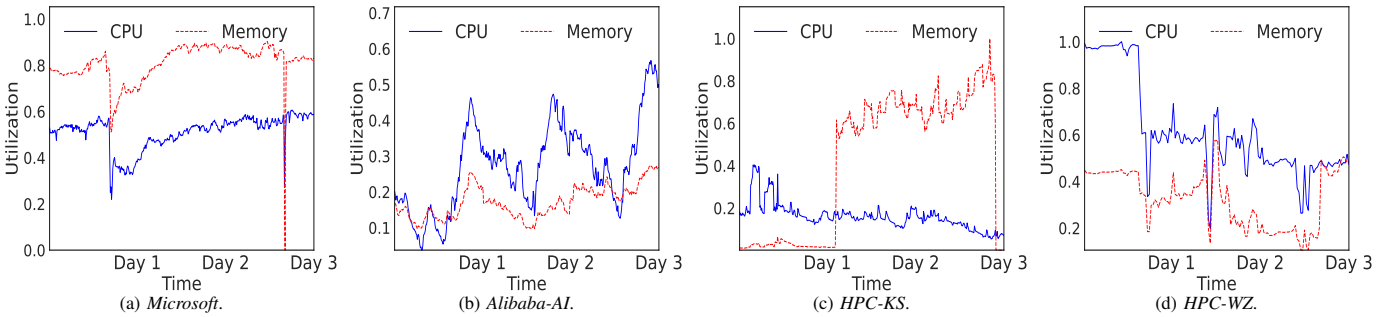
Data Heterogeneity in FL. Much research has focused on mitigating Non-IID issues via optimizing model update direction [13], [14]. Additionally, some methods utilize client selection to mitigate non-IID issues [17], [18]. From the perspective of data augmentation, some methods utilize shared GANs to synthesize data for knowledge distillation [15] or alleviating dataset shift [16]. However, these methods mostly focus on spatial data tasks, and are therefore unsuitable for the prediction of cloud workload time series data. Moreover, the sharing of workload features across cloud providers is against the privacy regulations of most cloud providers [7].

Federated GAN for Data Augmentation. The FL training of GAN models has been proposed for IID data augmentation. Hardy *et al.* [20] proposed a distributed GAN training method on clients with IID data. Rasouli *et al.* [21] extended FL to GAN by using two time-scale learning rates for the generator and discriminator of GAN. To address Non-IID issues, Guerraoui *et al.* [22] proposed to utilize the Kullback-Leibler distance between local and global label categories as the FL aggregation weight. Zhang *et al.* [23] proposed a FL framework for training GANs among clients with Non-IID data. Li *et al.* [24] proposed to weigh clients' GAN model updates by their spatial data synthesis quality. Ma *et al.* [25] proposed an unbiased Federated GAN scheme to mitigate Non-IID issues. However, these Federated GAN training methods are designed for spatial data tasks, thus cannot be applied for the FL training of time series GAN models due to the lack of a reliable metric for time series data synthesis quality assessment.

III. Preliminaries and Motivations

First, we present the detailed observations that motivate our system design, which are based on the following datasets:

- **Alibaba dataset [28]**. It is collected from a large cluster with over 4,000 machines in Alibaba data centers, spanning a period of 8 days in 2018, with a sampling period of ten seconds. It contains the logs of Alibaba's production workloads, which record workload information such as `cpu_util_percent`, `mem_util_precent`, etc.



(a) *Microsoft*. (b) *Alibaba-AI*. (c) *HPC-KS*. (d) *HPC-WZ*.

Fig. 1: Workloads of CPU and memory in four cloud providers.

- **Microsoft dataset [29]**. It is collected from Microsoft's Philly clusters with over hundreds of machines, spanning from August to December of 2017, with a sampling period of one minute. It contains the logs of >96,000 deep learning jobs, recording workload information such as `cpu_util`, `mem_util`, etc.
- **Google dataset [30]**. It is collected from Google production clusters with over 52,000 machines, which covers a 1-month time frame in 2011, with a sampling period of five minutes. It contains the logs of Google's production workloads, which record workload information such as `job_ID`, `mean_CPU_usage_rate`, etc.
- **Alibaba-AI dataset [31]**. It is collected from a large production cluster with over 6,500 GPUs (on ∼1,800 machines) in Alibaba Platform for Artificial Intelligence, spanning from July to August of 2020, with a sampling period of ten minutes. It records workload information such as `cpu_utilization`, `memory_utilization`, etc.
- **HPC datasets**. They are collected from three high performance computing cloud service centers (*HPC-KS*, *HPC-HF* and *HPC-WZ*) for academic research, where the *HPC-KS* dataset spans from January to February of 2022 with a sampling period of ten minutes, the *HPC-HF* dataset spans September in 2022 with a sampling period of five minutes, and the *HPC-WZ* dataset spans from January to October of 2022 with a sampling period of half an hour. They contain the logs of scientific computing, large-scale optimization or ML algorithms, which record workload information such as anonymized `job_cpu_time`, `job_mem_used`, etc.

Given the different information of the workloads, their temporal pattern may be highly heterogeneous, which will be illustrated with details in Section III-D.

A. Dynamic Time Warping

Given two time series $\mathbf{x} = \{x_1, \ldots, x_i, \ldots, x_n\}$ and $\mathbf{y} = \{y_1, \ldots, y_j, \ldots, y_m\}$, the DTW algorithm first calculates the distance matrix $\mathbf{D} = \{d(i,j)|1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$, where $d(i,j)$ is the Euclidean distance between x_i and y_j. The objective of DTW is to find the optimal path from $(1,1)$ to (n,m) with the minimum cost, which can be represented as the following recurrence and solved by dynamic programming:

$$\begin{aligned}\widetilde{d}(i,j) \leftarrow & d(i,j) + \min\{\widetilde{d}(i-1,j), \widetilde{d}(i,j-1), \\ & \widetilde{d}(i-1,j-1)\}\end{aligned} \tag{1}$$

where $\widetilde{d}(i,j)$ represents the cumulative distance of the optimal path from $(1,1)$ to (i,j). Thus, the cumulative distance is the sum of the distance between current elements (i.e., $d(i,j)$) and the minimum of the cumulative distances of the neighboring points (i.e., $\widetilde{d}(i-1,j)$, $\widetilde{d}(i,j-1)$ and $\widetilde{d}(i-1,j-1)$). Similarly, we can finally obtain the cumulative distance matrix $\widetilde{\mathbf{D}} = \{\widetilde{d}(i,j)|1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$, where $\widetilde{d}(n,m)$ is the DTW distance between $\mathbf{x}$ and $\mathbf{y}$. Although DTW reflects the heterogeneity between workload sequences, it has significant deficiencies when assessing data synthesis quality, as elaborated in Section III-E.

B. Federated GAN

Given K clients holding regular GANs as their local model, Federated GAN training aims to build a mapping function that maps a random noise sub-space $\mathbf{z}$ from randomized space $\mathbf{Z}$ into the overall workload data space of the clients $\bar{\mathbf{X}} = \cup_{k=1}^{K} \mathbf{x}_k$, where $\mathbf{x}_k$ is the workload sub-space of the k-th client. Thus, the objective function of the minimax game between the discriminator D and the generator G is:

$$\begin{aligned}\min_G \max_D V(G,D) = & \sum_{k=1}^{K} \alpha_k (\mathbb{E}_{x\sim p(\mathbf{x}_k)}[\log D_k(x)] + \\ & \mathbb{E}_{z\sim p(\mathbf{z})}[\log(1 - D_k(G_k(z)))]) \\ \text{s.t.} \sum_{k=1}^{K} \alpha_k = & 1\end{aligned} \tag{2}$$

where α_k represents the model update weight of the k-th client, and D_k and G_k are the discriminator and generator of the k-th client, respectively. $p(\mathbf{x}_k)$ is the local data distribution of the k-th client, while $p(\mathbf{z})$ is the noise distribution. Albeit effective in synthesizing IID data, existing Federated GAN training methods, lacking consideration of temporal features, are unsuitable for time series GAN model training.

C. Threat Model

We assume the FL server is honest-but-curious, which means it will follow the protocol of FL, but is curious about clients' private data and uploaded content. We also assume that the clients will use encryption (e.g. Homomorphic Encryption (HE)) [32] or privacy preservation techniques (e.g., Differential Privacy (DP)) [17], [33] to protect gradient information from inversion attacks [34]. The protection against other forms of malicious activities (e.g., collusion, poisoning, backdoor attacks) is not the focus of this work.

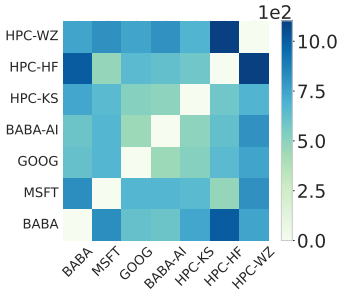


Fig. 2: Heat map matrix of DTW between CPU utilization time series.

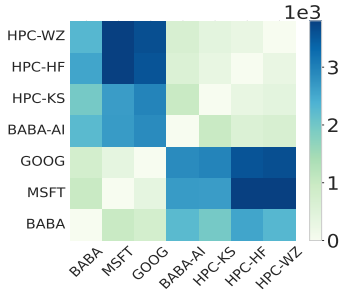
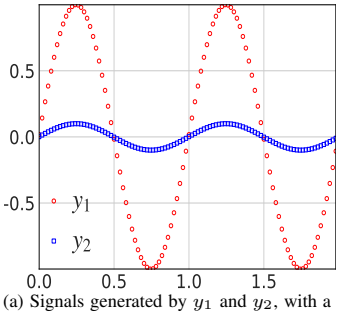
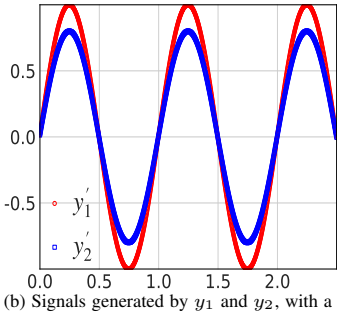


Fig. 3: Heat map matrix of DTW between memory utilization time series.



(a) Signals generated by y_1 and y_2, with a sampling frequency of 50 and an end point at $t = 2$.



(b) Signals generated by y_1 and y_2, with a sampling frequency of 500 and an end point at $t = 2.5$.

Fig. 4: Susceptibility of DTW to the length of time series.

D. Heterogeneous Temporal Features of Workload Data

From dataset introduction, we have known that the workload representations of different cloud providers differ significantly. For example, *Microsoft* uses `cpu_util` to directly represent CPU utilization of machines, while *Google* only records CPU utilization per job. Hence, we need to combine *Google*'s jobs by timestamp, and deduce the overall CPU utilization of all machines. In addition, even for the same resource, different cloud providers may use different names. For example, *Alibaba* only uses `mem_util_percent` to represent memory utilization, while *Google* uses three features: `canonical_memory_usage`, `assigned_memory_usage` and `maximum_memory_usage`. Therefore, to illustrate the heterogeneous temporal features of workloads, we first preprocess the datasets via aligning the representations of resources, removing redundant features and min-max normalization.

Finally, we obtain several workload time series of the seven cloud providers for further temporal feature heterogeneity analysis. Figure 1 illustrates the CPU utilization and memory utilization time series of four cloud providers over three days. It can be seen that the workload time series of *Alibaba-AI* follow obvious periodic patterns, while the workloads of the other cloud providers do not follow regular patterns. To confirm the data heterogeneity across the time series of all the seven cloud providers, we further calculate the DTW distances between the CPU utilization and memory utilization time series of every pair of them, which are illustrated as heat map matrices in Figure 2 and Figure 3, respectively. The darker the color of the square corresponding to two clients, the more dissimilar their workloads are. We can see that the workload time series between most pairs of cloud providers are highly dissimilar (i.e., Non-IID) in terms of DTW, especially for the CPU utilization time series.

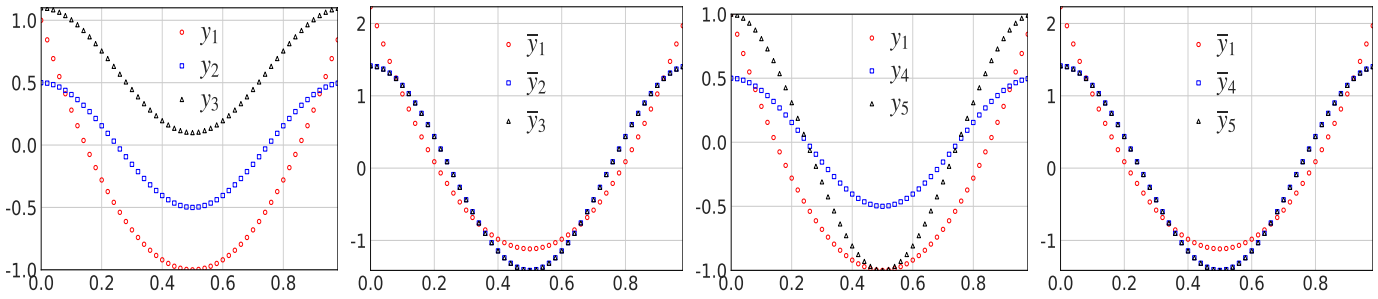
Challenge 1: given that the workload time series of different cloud providers are Non-IID in temporal features, while most existing FL solutions focus on spatial data, how to design a federated data augmentation method capable of synthesizing IID time series data?

E. Deficiency of DTW in Assessing Data Synthesis Quality

Although DTW can reflect the heterogeneity between workload sequences, it has two significant deficiencies when applied for assessing data synthesis quality: 1) DTW is susceptible to the length of time series, and 2) DTW overlooks the comparison of patterns between time series, which may make it unreliable for measuring the similarity of temporal patterns.

We use a toy example to illustrate the first deficiency. From the description of datasets, we can see that the workload time series of different cloud providers generally have various lengths due to different sampling frequencies and total time durations. Following these observations, we prepare two sets of signals that have different lengths due to the same factors. Specifically, the first set of signals are generated by functions $y_1 = \sin 2\pi t$ and $y_2 = 0.1 \sin 2\pi t$, with a sampling frequency of 50 and an end point at $t = 2$, as shown in Figure 4a. The second set of signals are generated by functions $y_1' = \sin 2\pi t$ and $y_2' = 0.8 \sin 2\pi t$, with a sampling frequency of 500 and an end point at $t = 2.5$, as shown in Figure 4b. Intuitively, the similarity between y_1' and y_2' is higher than that between y_1 and y_2 as the amplitude difference between y_1' and y_2' is obviously much lower than that between y_1 and y_2. However, the calculated DTW distance between y_1 and y_2 is 57.75, while the DTW distance between y_1' and y_2' is 72.58, which contradicts the straightforward intuition. This is primarily because that DTW measures the sum of absolute differences between the time series, so the measurement result is dependent on the length of the time series. Suppose the lengths of two sequences for DTW calculation are n and m. Possible solutions for fairly assessing data synthesis quality across cloud providers include normalizing DTW by $n + m$, $\max\{n, m\}$ or the warping path. In our case, since we set that the synthesized and original workload sequences have equal lengths (i.e., $n = m$), we simply normalize their DTW by $\max\{n, m\}$ (i.e., unit-length DTW distance). For example, the unit-length DTW distance between y_1' and y_2' is 0.058, while the unit-length DTW distance between y_1 and y_2 is 0.578, which is consistent with the intuitive observation.

For the second deficiency, we use two additional toy examples to illustrate the intrinsic reason. In the following, the sampling frequency and the end point of all signals are 50 and $t = 1$, respectively. Suppose we have a set of signals generated by functions $y_1 = 8(t - 0.5)^2 - 1$, $y_2 = 0.5 \cos 2\pi t$ and $y_3 = 0.5 \cos 2\pi t + 0.6$, as shown in Figure 5a. Intuitively, the similarity of temporal patterns between y_1 and y_2 is approximate as that between y_1 and y_3, as y_3 is actually y_2 displaced by 0.6 along the Y-axis. However, the DTW distance between y_1 and y_2 is 11.14, while the DTW distance



(a) Signals generated by y_1, y_2 and y_3. (b) $\bar{y}_2$ and $\bar{y}_3$ overlap after normalization. (c) Signals generated by y_1, y_4 and y_5. (d) $\bar{y}_4$ and $\bar{y}_5$ overlap after normalization.

Fig. 5: Unreliability of DTW in pattern similarity measurement of time series.

between y_1 and y_3 is 29.94. This is primarily because that as a distance metric, the DTW algorithm essentially computes the distance from the matching of similar elements between time series, but overlooks the comparison of patterns. Traditional methods [35]–[38] generally utilize the Z-score normalization to eliminate the influence of displacement: $\bar{y} = \frac{y-\mu}{\delta}$, where y and $\bar{y}$ are the original and normalized time series, respectively, μ and δ are the mean and standard deviation of y, respectively. For example, as illustrated in Figure 5b, the Z-normalized y_2 and y_3 (denoted as $\bar{y}_2$ and $\bar{y}_3$) completely overlap, of which DTW distances to y_1 are both 10.78.

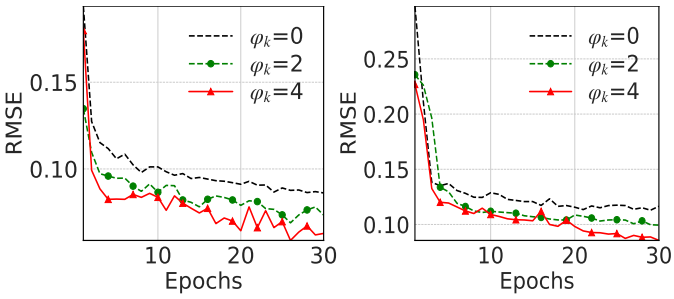
However, Z-score normalization does not work in all cases. Specifically, suppose in Figure 5c, we have another set of signals for pattern similarity comparison with y_1, which are generated by functions: $y_4 = 0.5\cos 2\pi t$ and $y_5 = \cos 2\pi t$. As we can see in Figure 5d, although the patterns of y_4 and y_5 are obviously quite different, the Z-normalized y_4 and y_5 (denoted as $\bar{y}_4$ and $\bar{y}_5$) also completely overlap, which is incorrect. This is mainly because that Z-score normalization scales the original time series amplitude by mean and standard deviation, which may cause time series with different patterns to be mistakenly scaled into the same waveform.

Challenge 2: given the deficiencies of DTW, how to design a fair data synthesis quality assessment method that is insusceptible to sequence length discrepancy across cloud providers and reliable for temporal pattern comparison?

F. Impact of Training Dataset Size on Workload Prediction Model Convergence

Generative Adversarial Active Learning (GAAL) is effective in augmenting data according to specific criteria of the learner [39]–[42], which is potentially suitable for the personalization of workload prediction model per cloud provider. Specifically, it uses GANs to synthesize informative data samples that are adapted to the workload prediction model. Then, the synthesized data is queried by a certain criterion and added back to the training dataset to update the workload prediction model. This protocol is executed iteratively until the predefined termination conditions are reached.

To guarantee the convergence of workload prediction model training, the ratio of synthesized data samples must be significantly higher than that of the local original data samples within the overall training dataset [43]. This means that the amount of data samples added to the training dataset per iteration must be comparable to that of the local original data samples. Under such cases, the learning rate may need to be adjusted accordingly with the amount of added data samples to maintain the stability of convergence.



(a) *Alibaba* (b) *Microsoft*

Fig. 6: Convergence speed under varying φ_k values in *Alibaba* and *Microsoft*.

To verify our conjecture, we conduct a GAAL experiment. We first use the CPU utilization and memory utilization workload data of all cloud providers to train a TimeGAN model [44] for workload data synthesis. Then, we add the synthesized data samples to the training dataset of each cloud provider as in GAAL, but with varying ratios of synthesized data samples. Specifically, the ratio of the k-th cloud provider is denoted as $\varphi_k = \frac{|G_k(z)|}{|\mathbf{x}_k|}$, where $G_k(z)$ represents the synthesized data samples, $\mathbf{x}_k$ represents the local original data samples, and $|\cdot|$ represents the amount of data samples. Finally, we train a GRU network under $\varphi_k = 0, 2, 4$, and calculate the Root Mean Square Error (RMSE $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{Predicted}_i - \text{Actual}_i)^2}$) of the prediction results. The GRU network consists of one input layer with the size of 2, one hidden layer with 512 hidden units, and one output layer with the size of 2. The number of training epochs is 30, and the learning rate is 0.001. The testing dataset is built by extracting a fixed portion of local workload data samples from each cloud provider.

Figure 6 illustrates the change of RMSE along with the training progress in two cloud providers under varying φ_k values. We can see that in *Alibaba*, the convergence curves under different φ_k values are quite divergent, which means that the model training on *Alibaba*'s dataset is sensitive to the change of φ_k. Moreover, when $\varphi_k = 4$, the RMSE curve in *Alibaba* oscillates significantly. This is primarily because that with the increase of synthesized data samples that follow the patterns of other cloud providers, the model parameter update direction differs from the training purely conducted on local original data samples. The static learning rate is probably

sub-optimal for the case with $\varphi_k = 4$, which causes the model training to repeatedly overshoot the optimal solution. In contrast, the convergence curves in *Microsoft* are much more stable and insensitive to the change of φ_k. Therefore, the learning rate does need to be adjusted according to the amount of added data samples to maintain the stability of workload prediction model convergence during training.

Challenge 3: given traditional approaches (e.g., [45], [46]) can only adjust learning rates according to the number of epochs or gradient information, how to design a training approach of prediction models that adjusts learning rate in accordance with the training dataset size?

IV. System Design

Suppose the collaborative training of workload prediction models involves K cloud providers (clients), which have highly Non-IID workload datasets. *P³Forecast* consists of the following three stages as shown in Figure 7:

1. Workload Data Synthesis Quality Assessment (Section IV-A). First, given the local workload time series and a synthesized workload time series, each client calculates the distance matrix, which reflects both the temporal value discrepancy and pattern similarity between the two time series. Then, each client obtains its cumulative distance matrix via dynamic programming as in Equation (1). Finally, by normalizing the minimum cumulative distance into the unit-length DTW distance, each client obtains its *pattern-aware DTW* as the data synthesis quality assessment result.

2. Federated GAN Training based on Data Synthesis Quality (Section IV-B). By using the *pattern-aware DTW* as the weight of model updates from the cloud providers, the FL server aggregates the generators and discriminators to train the global GAN model, which captures the workload features of all the clients. Upon the completion of FL training, the FL server sends the trained global generator to the clients for workload data synthesis.

3. Post-Training of Local Workload Prediction Models (Section IV-C). After receiving the trained generator, each client first uses it to synthesize data samples, and extracts the most informative ones by a query mechanism. Then, each client adds the queried data samples back to its training dataset, and adjusts the learning rate in accordance with the training dataset size. Upon the completion of the post-training, each client obtains its personalized workload prediction model.

A. Workload Data Synthesis Quality Assessment

Given the effectiveness of DTW in measuring time series heterogeneity and its deficiencies illustrated in Section III-E, we aim to make it insusceptible to time series length and reliable for the comparison of temporal patterns. Considering that the first-order difference reflects the volatility of a time series $\mathbf{x}$ (defined as $\nabla x_i = x_i - x_{i-1}$) [47], which measures the magnitude of growth or decline at each time point (i.e., change pattern), we incorporate it in DTW for data synthesis quality assessment, which is called the *pattern-aware DTW*.

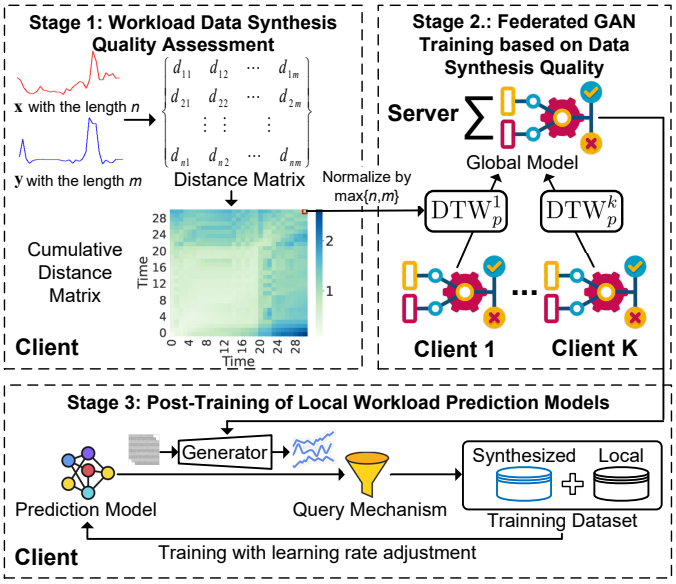


Fig. 7: Overall architecture of *P³Forecast*.

Specifically, given two time series $\mathbf{x}$ and $\mathbf{y}$ with the lengths of n and m, respectively. Suppose the first-order differences of two time series $\mathbf{x}$ and $\mathbf{y}$ at the i-th and the j-th time points are ∇x_i and ∇y_j, respectively. Instead of directly calculating the Euclidean distance between x_i and y_j as the element $d(i,j)$ in the distance matrix $\mathbf{D}$ (Section III-A), we additionally calculate the Euclidean distance between ∇x_i and ∇y_j, and use it as $d(i,j)$ if the patterns of $\mathbf{x}$ and $\mathbf{y}$ are comparable at the respective time points. By saying comparable, we mean ∇x_i and ∇y_j have identical signs (i.e., $\nabla x_i \nabla y_j > 0$) or their absolute difference is smaller than a threshold (i.e., $|\nabla x_i - \nabla y_j| < \epsilon$). The rationale is that when ∇x_i and ∇y_j are comparable, we prefer $\mathbf{D}$ to reflect the similarity between patterns rather than the distance between points in the time series. Thus, in *pattern-aware DTW*, $d(i,j)$ is calcualted as:

$$d(i,j) = \begin{cases} |\nabla x_i - \nabla y_j|, & \nabla x_i \nabla y_j > 0 \vee |\nabla x_i - \nabla y_j| < \epsilon \\ |x_i - y_j|, & \text{otherwise.} \end{cases} \quad (3)$$

Note that the threshold ϵ (e.g., 0.001) is set to ensure that ∇x_i and ∇y_j are sufficiently approximate when they have different signs, which can be adjusted according to specific workload data synthesis quality assessment requirements. Then, we obtain the minimum cumulative distance between $\mathbf{x}$ and $\mathbf{y}$, i.e., $\widetilde{d}(n,m)$, via dynamic programming as in Equation (1). Finally, as illustrated in Section III-E, to make *pattern-aware DTW* insusceptible to time series length, the distance between $\mathbf{x}$ and $\mathbf{y}$ is calculated as $\mathrm{DTW}_p(\mathbf{x},\mathbf{y}) = \frac{\widetilde{d}(n,m)}{\max\{n,m\}}$, i.e., unit-length distance. Thus, $\mathrm{DTW}_p(\mathbf{x},\mathbf{y})$ reflects both their distance between temporal values and similarity between temporal patterns regardless of their lengths. The workflow of calculating the *pattern-aware DTW* distance is illustrated in "Stage 1" of Figure 7, where the heatmap shows the cumulative distance matrix between the original workload sequence (red) and the synthesized one (blue), which have equal lengths. The element highlighted with the red square is $\widetilde{d}(n,m)$. After normalizing it by $\max\{n,m\}$, it represents the

pattern-aware DTW distance between the two sequences.

B. Federated GAN Training based on Data Synthesis Quality

According to Section III-D, we have known that the workload data in different cloud providers is Non-IID. If we directly utilize the data for FL training of workload prediction models, the obtained models may suffer from severe performance degradation [12], [13]. Instead, we let the clients collaboratively train a GAN model via Federated GAN training as described in Section III-B, which can be utilized by each client to synthesize its own IID training data.

However, the traditional Federated GAN training methods cannot adequately attend to the temporal features unique to time-series data. Considering that the *pattern-aware DTW* distance between the workload data of each cloud provider (denoted as $\mathbf{x}_k$) and the data synthesized by the local GAN model at the t-th training round (denoted as $G_k^t(z)$), i.e., $\text{DTW}_p(\mathbf{x}_k, G_k^t(z))$, reflects their discrepancy in temporal features (i.e., data synthesis quality), we use it to weigh the model updates in Federated GAN training. Specifically, as $\text{DTW}_p(\mathbf{x}_k, G_k^t(z))$ actually measures the unit-length distance (i.e., the larger it is, the less similar $\mathbf{x}_k$ is to $G_k^t(z)$), we take its reciprocal $\text{DTW}_p^{-1}(\mathbf{x}_k, G_k^t(z))$ as their similarity. Moreover, as the *pattern-aware DTW* distances of the K cloud providers are measured on different scales and independent of each other, we let the aggregation server normalize them as:

$$\alpha_k = \frac{\text{DTW}_p^{-1}(\mathbf{x}_k, G_k^t(z))}{\sum_{k=1}^{K} \text{DTW}_p^{-1}(\mathbf{x}_k, G_k^t(z))}. \tag{4}$$

Additionally, to avoid the vanishing gradient problem (i.e., when the discriminator is much more powerful than the generator, the generator always fails to produce convincing data samples [22]), we simultaneously aggregate the local discriminators and generators of the K cloud providers during FL training, ensuring that they have the same power in confronting each other. By following the objective defined in Equation (2), the aggregation of local discriminators and generators at the t-th round can be described as:

$$\begin{cases} D^t \leftarrow \sum_{k=1}^{K} \alpha_k D_k^t \\ G^t \leftarrow \sum_{k=1}^{K} \alpha_k G_k^t \end{cases} \tag{5}$$

After aggregation, the global generator G is optimal if and only if it has learned the temporal features of all data samples. That is, the data space of G covers the overall workload data space of all the cloud providers (i.e., $\bar{\mathbf{X}} = \cup_{k=1}^{K} \mathbf{x}_k$). Note that since the aggregation step in Equation (5) is independent of model specifics, it can be actually applied to any existing GAN model architectures. The workflow of our proposed Federated GAN training method is illustrated in "Stage 2" of Figure 7.

C. Post-Training of Local Workload Prediction Models

Although the global generator is trained with all the workload data across clients, each client only needs the most informative synthesized data samples that can augment its training dataset. To realize this goal, we develop a post-training method of local workload prediction models by following the rationale of GAAL [39]–[42]. The method utilizes the trained generator to synthesize data samples, and applies a novel query mechanism to filter out the ones that are detrimental to the training of the workload prediction model per client.

Specifically, suppose $\mathbf{X}_k$ represents the overall training dataset of the k-th cloud provider, which consists of the data synthesized by the trained global generator ($G(z)$) and the cloud provider's local original data ($\mathbf{x_k}$). In the first round of post-training, each cloud provider trains the prediction model with only local workload data to mitigate or prevent overfitting. For the subsequent rounds, each cloud provider first uses the trained global generator to synthesize data $G(z)$, which has the same length as $\mathbf{x_k}$. Then, each cloud provider tests the newly synthesized $G(z)$ with its workload prediction model, calculates the RMSE of each data sample, and queries high-quality synthesized data samples by the following criterion:

$$\mathbf{Q}_k = \left\{ G(z) \;\middle|\; \begin{array}{l} \text{RMSE}(G(z), \widehat{G}(z)) \leqslant \text{RMSE}(\mathbf{X}_k, \widehat{\mathbf{X}}_k) \\ \text{and } \mathbf{x_k} \subsetneq \mathbf{X}_k \text{ or } \mathbf{x_k} = \mathbf{X}_k \end{array} \right\} \tag{6}$$

where $\mathbf{Q}_k$ is the query result, $\widehat{G}(z)$ and $\widehat{\mathbf{X}}_k$ are the outputs of the local workload prediction model with $G(z)$ and $\mathbf{X}_k$ as the inputs, respectively. The criterion means that the queried data samples must either improve the training data quality in terms of RMSE (i.e., $\text{RMSE}(G(z), \widehat{G}(z)) \leqslant \text{RMSE}(\mathbf{X}_k, \widehat{\mathbf{X}}_k)$ and $\mathbf{x_k} \subsetneq \mathbf{X}_k$), or augment the training dataset for the first time (i.e., $\mathbf{x_k} = \mathbf{X_k}$). Next, each cloud provider merges the queried high-quality synthesized workload data into the training dataset as $\mathbf{X}_k = \mathbf{X}_k \cup \mathbf{Q}_k$.

With the progress of post-training, the size of $\mathbf{X}_k$ will gradually grow. From Section III-F, we have known that the training dataset size has varying impact on model convergence for different cloud providers. Therefore, we design a personalized strategy that adjusts learning rate in accordance with the training dataset size. On the one hand, we expect the learning rate to gradually decrease as the training dataset size increases, in order to maintain the stability of model convergence. On the other hand, we don't expect the learning rate to decrease too fast, as an excessively low learning rate could lead to model under-fitting. Therefore, given the ratio between the amount of synthesized data samples and the amount of local data samples of the k-th cloud provider (i.e., φ_k), we use an exponential function of φ_k to scale the learning rate η_k as:

$$\eta_k = \eta^0 e^{-\mu_k \varphi_k} \tag{7}$$

where η^0 is the initial learning rate, and μ_k is the constant controlling the impact of the training dataset size on the learning rate. To determine the optimal μ_k for each cloud provider, we vary μ_k within a range (e.g., [0, 0.3]), and post-train workload prediction models under each μ_k value per cloud provider. Then, the optimal μ_k yielding the lowest RMSE is selected for each cloud provider. The post-training workflow of local workload prediction models is shown in "Stage 3" of Figure 7.

It is worth noticing that unlike most existing FL personalization methods that require consistent model architectures across clients, *P³Forecast* grants clients high flexibility in independently specifying prediction model architectures (e.g., LSTM,

Algorithm 1: Workflow of *P³Forecast*.

1 **Server executes:**
2 Initialize the global GAN model D^0 and G^0;
3 **for** *each round* $t = 1, 2, \ldots$ **do**
4 **for** $k = 1$ **to** K **in parallel do**
5 $D_k^t, G_k^t, \text{DTW}_p(\mathbf{x}_k, G_k^t(z))$ = ClientUpdate(D^{t-1}, G^{t-1});
6 **for** $k = 1$ **to** K **do**
7 $\alpha_k = \frac{\text{DTW}_p^{-1}(\mathbf{x}_k, G_k^t(z))}{\sum_{k=1}^{K} \text{DTW}_p^{-1}(\mathbf{x}_k, G_k^t(z))}$;
8 Aggregate local GAN models by Equation (5);
9 **for** $k = 1$ **to** K **in parallel do**
10 ClientPostTraining(G);
11 **Clients execute:**
12 Initialize local prediction model;
13 ClientUpdate(D^{t-1}, G^{t-1}):
14 Set $D_k^{t-1} = D^{t-1}$, $G_k^{t-1} = G^{t-1}$;
15 **for** *each epoch* $e = 1, 2, \ldots$ **do**
16 $D_k^t, G_k^t \leftarrow$ Update local GAN model with $\mathbf{x}_k$;
17 Synthesize workload data $G_k^t(z)$;
18 Calculate $\text{DTW}_p(\mathbf{x}_k, G_k^t(z))$ as in Section IV-A;
19 **return** $D_k^t, G_k^t, \text{DTW}_p(\mathbf{x}_k, G_k^t(z))$;
20 ClientPostTraining(G):
21 **for** *each round* $t = 1, 2, \ldots$ **do**
22 Synthesize data with $|G(z)| = |\mathbf{x_k}|$;
23 $\mathbf{Q}_k \leftarrow$ Query $G(z)$ by Equation (6) ; $\mathbf{X}_k \leftarrow \mathbf{X}_k \cup \mathbf{Q}_k$; $\eta_k \leftarrow$ Update η_k as Equation (7); Train workload prediction model with $\mathbf{X}_k$ using η_k;

Transformers, etc.) according to their own needs, which is especially suitable for cloud providers with heterogeneous model preferences and data augmentation preferences.

Finally, the workflow of *P³Forecast* is summarized as Algorithm 1. At Lines 2 and 12, the server and clients initialize the global GAN model and local prediction models, respectively. At Lines 4-5 and Lines 13-19, each client updates its GAN model with its original workload data $\mathbf{x}_k$, synthesizes data $G_k^t(z)$, and computes the *pattern-aware DTW* distance $\text{DTW}_p(\mathbf{x}_k, G_k^t(z))$ as in Section IV-A. At Lines 6-8, the server aggregates the clients' local GAN models as in Section IV-B. At Lines 9-10 and Lines 20-24, each client uses the trained global generator to augment its training dataset and post-trains its workload prediction model as in Section IV-C.

D. Security Analysis

The previous methods using shared GANs for FL training data augmentation [15], [16], [19] require participants to share the base noise that covers all the heterogeneous data features across clients, which may cause privacy leakage and violate the privacy regulations of most cloud providers [7]. In contrast, the clients in *P³Forecast* only upload GAN model parameters and the data synthesis quality assessment result DTW_p to the FL server for normalization and model aggregation. For the former, *P³Forecast* can refer to various privacy preservation schemes (e.g., HE, DP) to prevent adversaries from accessing the actual model parameters. For the latter, DTW_p is a scalar value and does not contain any privacy-sensitive information. Therefore, *P³Forecast* can ensure the privacy protection of clients while allowing the collaborative training of GAN models for IID training data augmentation, and personalization of local workload prediction models.

V. Performance Evaluation

A. Comparison Methods

For the evaluation of workload prediction accuracy, we compare *P³Forecast* with the vanilla *FedAvg* framework [10], a representative FL framework for tackling heterogeneity [14] (denoted as *FedProx*), and a GRU-based cloud workload prediction model (denoted as *GRU*) [6], which serves as the baseline. Specifically, *FedAvg* weighs and aggregates clients' model updates by data quantity. *FedProx* introduces a proximal term to penalize the deviation between local models and the global model. *GRU* exploits a sliding window and a revised GRU neural network for cloud workload prediction. Note that *GRU* trains workload prediction models for each cloud provider via solely using its local data.

For convergence analysis, we compare *P³Forecast* with a popular GAN variant (denoted as *IFL-GAN*) [24]. and the vanilla FL GAN (denoted as *Baseline*) [20]. Specifically, *IFL-GAN* aims to train a globally shared GAN model via aggregating the generators' model updates trained on the clients' local data, of which data synthesis quality is assessed in terms of the Maximum Mean Discrepancy (MMD). In *Baseline*, the GAN model parameters are directly aggregated by the FedAvg algorithm [10], of which weights are determined according to the quantity of data samples per client. To illustrate the effectiveness of our proposed *pattern-aware DTW* distance, we also evaluate the convergence performance of a variant of *P³Forecast*, which simply uses the DTW distance in FL model aggregation (denoted as *P³Forecast-DTW*).

B. Experimental Settings

Datasets. The datasets for experiments are the same as in Section III. Specifically, we use the first 70% as the local training dataset per cloud provider. Since there is no available cross-cloud workload dataset covering various types of workload patterns, we extract the remaining 30% workload data from each cloud provider to build the testing dataset.

Models. For fairness, we select the TimeGAN model architecture [44] for all the comparison methods that use GAN for workload data synthesis. Specifically, both the embedding network and the generator use a GRU network consisting of one input layer with the size of 2, three hidden layers with 256 hidden units and one output layer with the size of 256. The recovery network uses a GRU network consisting of one input layer with the size of 256, three hidden layers with 256 hidden units and one output layer with the size of 2. The discriminator uses a GRU network consisting of one input layer with the size of 256, three hidden layers with 256 hidden units and one output layer with the size of 1. Moreover, considering that the comparison methods do not grant clients the flexibility of specifying their own workload

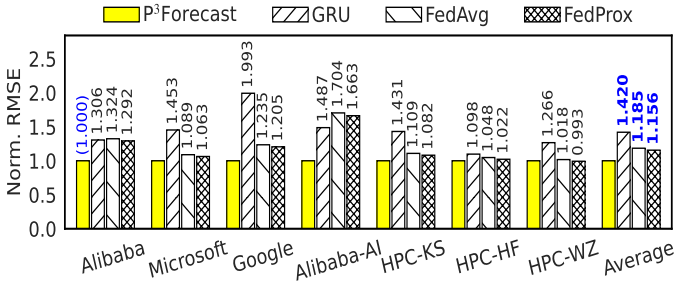


Fig. 8: Workload prediction accuracy on different cloud providers (normalized by the RMSE of *P³Forecast*).

prediction models as in *P³Forecast*, we let all cloud providers use the same GRU network as the prediction model for fair evaluation, which consists of one input layer with the size of 2, one hidden layer with 512 hidden units, and one output layer with the size of 2. We set ϵ in Equation (3) to 0.001.

Implementation. We conduct evaluations on a Linux server with 2 GeForce RTX 3090 GPUs, 1 Intel Xeon CPU, and 256 GB memory. We implement our framework with PyTorch. The FL clients are *Alibaba*, *Microsoft*, *Google*, *Alibaba-AI*, *HPC-KS*, *HPC-HF* and *HPC-WZ*.

C. Experimental Results

1) Workload Prediction Accuracy: In this section, we evaluate the performance of different methods in terms of workload prediction accuracy. Specifically, in *GRU*, the cloud providers separately train their prediction models with their isolated workload data for 30 epochs. In *FedAvg* and *FedProx*, the global FL training is conducted for 6 rounds with 5 local training epochs per round. The hyper-parameters of these comparison methods are fine-tuned to reflect their optimal performance. In *P³Forecast*, the global FL training of the TimeGAN model is conducted for 10 rounds with 500 local training epochs per round. The post-training of the local prediction model is conducted for 6 rounds, during which the local prediction model is trained for 5 epochs per round with the workload data synthesized by the trained TimeGAN model. In this way, we heuristically set $\max(\varphi_k) = 5$ as in [43] for the balance between model performance and data augmentation computation overhead, and leave the sensitivity analysis of varying φ_k to future extensions. In all the methods, the batch sizes for the training of the GRU and TimeGAN models are 128, while the lengths of observation time step for the two models are 64 and 65, respectively. In the TimeGAN model, the first 64 time steps are used as the input and the last 1 time step is used as the output. We employ the Adam optimizer [46] for both the GRU and TimeGAN models, using an initial learning rate of 0.001 and beta values of 0.9 and 0.999, respectively. μ_k is uniformly set to 0.1 in this experiment.

For each cloud provider, we measure the average RMSEs of workload prediction results on CPU utilization and memory utilization in different methods, which are illustrated in Figure 8. For the convenience of comparison, all RMSEs are normalized by the RMSE of *P³Forecast*. The overall performance across all the cloud providers generally follows: *P³Forecast* <*FedProx*<*FedAvg*<*GRU*. We can observe that all the FL-based methods (i.e., *FedAvg*, *FedProx*, and *P³Forecast*) outperform *GRU* in almost all the cloud providers. This is primarily because that in *GRU*, the prediction models are separately trained per cloud provider with its isolated workload dataset. Thus, on the testing dataset with various workload patterns, the trained models of most cloud providers cannot achieve comparable performance as the ones collaboratively trained with the FL algorithms. Moreover, we can notice that the performance of *GRU* is especially bad on *Google*, which is due to the singular workload patterns of *Google* compared to the other cloud providers. The prediction model solely trained on *Google*'s data has insufficient knowledge on the workload patterns of the other cloud providers, and hence result in the model's inferior performance on the testing dataset.

TABLE I: Change of RMSEs under various μ_k values.

Cloud Providers	μ_k				
	0	0.05	0.1	0.2	0.3
Alibaba	0.0681	0.0711	**0.0660**	0.0669	0.0674
Microsoft	0.0844	0.0829	**0.0802**	0.0835	0.0864
Google	0.0677	**0.0649**	0.0707	0.0668	0.0714
Alibaba-AI	0.0574	0.0591	0.0513	**0.0491**	0.0524
HPC-KS	0.0788	**0.0747**	0.0788	0.0815	0.0826
HPC-HF	0.0879	0.0863	**0.0834**	0.0870	0.0955
HPC-WZ	0.0926	**0.0812**	0.0859	0.0885	0.0922
Average	0.0767	0.0743	**0.0738**	0.0748	0.0783

We can also observe that the RMSEs of workload prediction results in *FedAvg* and *FedProx* are quite approximate. This is mostly because that in these methods, the prediction models trained via FL are not personalized according to the heterogeneity degree of local data. For the cloud providers that already have a rich variety of workload patterns, the patterns learned from the others may actually deteriorate their model performance (e.g., *Alibaba-AI*).

In contrast, *P³Forecast* is 15.6%, 18.5% and 42.0% more accurate in average over all the cloud providers when compared with *FedProx*, *FedAvg* and *GRU*, respectively. For the cloud providers with relatively more singular workload patterns, (e.g., *Google*), *P³Forecast* can achieve as much as 99.3% improvement in accuracy compared to *GRU*. The significant performance improvement can be attributed to our proposed Federated GAN training method based on workload data synthesis quality assessment, which ensures the capture of cross-cloud workload patterns in data synthesis, and post-training of workload prediction models, which ensures the personalized augmentation of training data according to the heterogeneity degree of cloud providers' local data. As a result, *P³Forecast* can effectively mitigate data heterogeneity while enjoying the workload patterns contributed across the clouds.

2) Effectiveness of Learning Rate Adjustment: Recall that μ_k controls the impact of the training dataset size on the learning rate in Equation (7). Considering that in Section III-F, we have known that different cloud providers have various sensitivities to the change of φ_k, we conjecture that the optimal μ_k values also vary on the cloud providers during post-

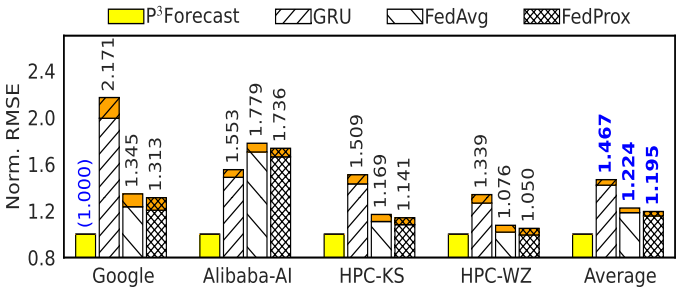


Fig. 9: Workload prediction accuracy using the optimal μ_k per cloud provider.

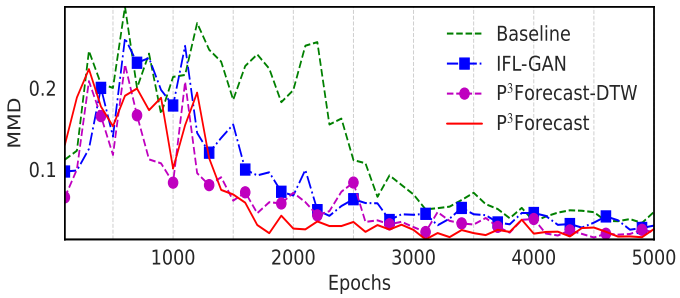


Fig. 10: Convergence performance of different methods.

training. To confirm this, we investigate the impact of various μ_k values on model performance in *P³Forecast*. Specifically, we set μ_k to 0, 0.05, 0.1, 0.2 and 0.3 during the post-training of workload prediction models for each cloud provider, and measure the RMSEs of prediction results, which are shown in Table I. The optimal μ_k values resulting in the lowest RMSE are highlighted in bold.

First, we can observe that when $\mu_k = 0$ (i.e., the learning rate is fixed to η^0 in Equation (7)), none of the cloud providers can harvest the prediction results with the lowest RMSE. Some cloud providers even suffer from >14% increase in RMSE (e.g., *Alibaba-AI* and *HPC-WZ*). This means that the adjustment of the learning rate in accordance with the training dataset size is necessary. We can also notice that when μ_k is significantly large (e.g., $\mu_k > 0.2$), the RMSEs of all the cloud providers deteriorate as well. This is mainly because that a larger μ_k value causes the learning rate to decrease sharply as the training dataset size grows, leading to model underfitting.

Due to the heterogeneity of workload data, the cloud providers have varying optimal μ_k values, which concentrate between 0.05, 0.1 and 0.2. To illustrate the effectiveness of the proposed learning rate adjustment method, we further compare the workload prediction accuracy of *P³Forecast*, which uses the optimal μ_k per cloud provider in post-training, with the others. As 0.1 is already the optimal μ_k value for *Alibaba*, *Microsoft* and *HPC-HF*, we only update prediction RMSEs of the other cloud providers. Figure 9 illustrates their updated prediction RMSEs, normalized by that of *P³Forecast*. Compared with Figure 8, the improvements resulted from using the optimal μ_k values are marked with orange segments. We can see that after adjusting the μ_k values in *P³Forecast*, all the cloud providers enjoy varying degrees of improvement on prediction accuracy. Especially for *HPC-WZ*, where *P³Forecast* underperforms *FedProx* when μ_k is uniformly set to 0.1, but outperforms the others after adjusting μ_k to 0.05. In summary, when using the optimal μ_k value per cloud provider in post-training, *P³Forecast* is 19.5%, 22.4% and 46.7% more accurate in average over all the cloud providers when compared with *FedProx*, *FedAvg* and *GRU*, respectively.

3) Convergence Analysis of Federated GAN Training: We further evaluate the convergence performance and data synthesis quality of the methods. We use MMD as the metric of data synthesis quality, of which change along with the training progress can effectively reflect how well the methods converge [48]–[50]. Specifically, the MMD between the local data x and the synthesized data $G(z)$ is calculated as:

$$\text{MMD} = \sup \left\| \mathbb{E}_{x \sim p(\mathbf{x}_k)}[\phi(x)] - \mathbb{E}_{z \sim p(\mathbf{z})}[\phi(G(z))] \right\|^2, \quad (8)$$

where $\phi(\cdot)$ is the Gaussian kernel function. A lower MMD indicates a better data synthesis quality. During the training of the Federated GAN models in each method, we record the MMD between x and $G(z)$ every 100 epochs, which is illustrated in Figure 10. We can see that in the first 1300 epochs, *IFL-GAN*, *P³Forecast* and *P³Forecast-DTW* converge conspicuously faster than *Baseline*, which confirms the effectiveness of taking into account data synthesis quality in the weighing and aggregation of model updates in Federated GAN training. However, after 1300 epochs, *P³Forecast* and *P³Forecast-DTW* converge faster and finally stabilize at better MMD scores than *IFL-GAN*. This is primarily because that although MMD is effective in measuring the dissimilarity between sample distributions, it neglects the temporal features unique to time-series data. What's more, we can also notice that *P³Forecast* quickly stabilizes at around 1800 epochs, while *P³Forecast-DTW* has to spend more than 4000 epochs to achieve a comparable MMD level. This can be attributed to our proposed *pattern-aware DTW*, which enables the trained GAN model to synthesize data that is similar to the real one not only in temporal distance, but also in patterns.

VI. Conclusion

In this paper, we have extensively demonstrated the high Non-IID nature of cloud workloads and the deficiencies of existing methods in realizing collaborative training of workload prediction models. To address the issues, we proposed *P³Forecast*, a personalized privacy-preserving cloud workload prediction framework that extends DTW for data synthesis quality assessment, adopts Federated GAN for training data augmentation across cloud providers, and personalizes the workload prediction model per cloud provider via post-training. Our experiments conducted on real-world workloads showed that compared with the state-of-the-art, *P³Forecast* drastically improved workload prediction accuracy by 19.5%-46.7% in average over all cloud providers, while ensuring the fastest convergence in Federated GAN training. In the future, we plan to focus on adaptive adjustment of μ_k and φ_k via Bayesian optimization or Reinforcement Learning to further increase the prediction accuracy of workloads.

Acknowledgements

This work was supported by the National Key R&D Program of China (Grant No. 2022YFB4500800), the National

Natural Science Foundation of China (Grants No. 62103325 and 62202216), the Shaanxi High-Level Talent Program (Grant No. 2021QCYRC4-26), the Guangdong Basic and Applied Basic Research Foundation Grant (No. 2023A1515010244), and the Shenzhen Science and Technology Program (Grant No. 20231121101752002). Corresponding author: Li Yan.

REFERENCES

[1] I. K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Forecasting cloud application workloads with cloudinsight for predictive resource management," *IEEE TCC*, vol. 10, no. 3, 2020.

[2] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *Proc. of NSDI*, 2022.

[3] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' QoS," *IEEE TCC*, vol. 3, no. 4, 2015.

[4] S. Ouhame, Y. Hadi, and A. Ullah, "An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model," *Neural Computing and Applications*, vol. 33, 2021.

[5] Z. Chen, J. Hu, G. Min, A. Y. Zomaya, and T. El-Ghazawi, "Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning," *IEEE TPDS*, vol. 31, no. 4, 2019.

[6] M. Xu, C. Song, H. Wu, S. S. Gill, K. Ye, and C. Xu, "esdnn: deep neural network based multivariate workload prediction in cloud computing environments," *ACM TOIT*, vol. 22, no. 3, 2022.

[7] Z. Yang, Z. Wu, M. Luo, W.-L. Chiang, R. Bhardwaj, W. Kwon, S. Zhuang, F. S. Luan, G. Mittal, S. Shenker, and I. Stoica, "SkyPilot: An intercloud broker for sky computing," in *Proc. of NSDI*, 2023.

[8] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, 2018.

[9] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. of S&P*, 2019.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTATS*, 2017.

[11] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*. National Institute of Standards and Technology, 2011.

[12] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. of ICLR*, 2020.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. of ICML*, 2020.

[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. of MLSys*, 2020.

[15] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. of ICML*, 2021.

[16] Z. Tang, Y. Zhang, S. Shi, X. He, B. Han, and X. Chu, "Virtual homogeneity learning: Defending against data heterogeneity in federated learning," in *Proc. of ICML*, 2022.

[17] J. Wolfrath, N. Sreekumar, D. Kumar, Y. Wang, and A. Chandra, "Haccs: Heterogeneity-aware clustered client selection for accelerated federated learning," in *Proc. of IPDPS*, 2022.

[18] R. Zhou, J. Yu, R. Wang, B. Li, J. Jiang, and L. Wu, "A reinforcement learning approach for minimizing job completion time in clustered federated learning," in *Proc. of INFOCOM*, 2023.

[19] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[20] C. Hardy, E. Le Merrer, and B. Sericola, "Md-gan: Multi-discriminator generative adversarial networks for distributed datasets," in *Proc. of IPDPS*, 2019.

[21] M. Rasouli, T. Sun, and R. Rajagopal, "Fedgan: Federated generative adversarial networks for distributed data," *arXiv preprint arXiv:2006.07228*, 2020.

[22] R. Guerraoui, A. Guirguis, A.-M. Kermarrec, and E. L. Merrer, "Fegan: Scaling distributed gans," in *Proc. of Middleware*, 2020.

[23] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A novel federated learning scheme for generative adversarial networks," *IEEE TMC*, vol. 23, no. 5, 2024.

[24] W. Li, J. Chen, Z. Wang, Z. Shen, C. Ma, and X. Cui, "Ifl-gan: Improved federated learning generative adversarial network with maximum mean discrepancy model aggregation," *IEEE TNNLS*, vol. 34, no. 12, 2023.

[25] Z. Ma, Y. Liu, Y. Miao, G. Xu, X. Liu, J. Ma, and R. H. Deng, "Flgan: Gan-based unbiased federated learning under non-iid settings," *IEEE TKDE*, vol. 36, no. 4, 2024.

[26] E. Brophy, "Synthesis of dependent multichannel ecg using generative adversarial networks," in *Proc. of CIKM*, 2020.

[27] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A systematic literature review," *ACM Computing Surveys*, vol. 55, no. 10, 2023.

[28] "Alibaba cluster data v2018," https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2018, 2023, accessed in October, 2024.

[29] "Microsoft cluster data," https://github.com/msr-fiddle/philly-traces, 2023, accessed in October, 2024.

[30] "Google cluster data," https://github.com/google/cluster-data/blob/master, 2014, accessed in October, 2024.

[31] "Alibaba cluster data v2020," https://github.com/alibaba/clusterdata/tree/master, 2023, accessed in October, 2024.

[32] J. Han and L. Yan, "Adaptive batch homomorphic encryption for joint federated learning in cross-device scenarios," *IEEE IoT-J*, vol. 11, no. 6, 2023.

[33] Q. He, Z. Feng, H. Fang, X. Wang, L. Zhao, Y. Yao, and K. Yu, "A blockchain-based scheme for secure data offloading in healthcare with deep reinforcement learning," *IEEE/ACM ToN*, vol. 32, no. 1, 2023.

[34] K. Yue, R. Jin, C.-W. Wong, D. Baron, and H. Dai, "Gradient obfuscation gives a false sense of security in federated learning," in *Proc. of USENIX Security*, 2023.

[35] Y. Hao, Y. Chen, J. Zakaria, B. Hu, T. Rakthanmanon, and E. Keogh, "Towards never-ending learning from time series streams," in *Proc. of SIGKDD*, 2013.

[36] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. of SIGKDD*, 2012.

[37] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance measures for effective clustering of arima time-series," in *Proc. of ICDM*, 2001.

[38] G. E. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proc. of ICDM*, 2011.

[39] J.-J. Zhu and J. Bento, "Generative adversarial active learning," *arXiv preprint arXiv:1702.07956*, 2017.

[40] C. Nielsen and M. Okoniewski, "Gan data augmentation through active learning inspired sample acquisition," in *Proc. of CVPR Workshop*, 2019.

[41] C. Mayer and R. Timofte, "Adversarial sampling for active learning," in *Proc. of WACV*, 2020.

[42] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, "Generative adversarial active learning for unsupervised outlier detection," *IEEE TKDE*, vol. 32, no. 8, 2020.

[43] X. Cao, G. Sun, H. Yu, and M. Guizani, "Perfed-gan: Personalized federated learning via generative adversarial networks," *IEEE IoT-J*, vol. 10, no. 5, 2023.

[44] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Proc. of NeurIPS*, 2019.

[45] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. of ICLR*, 2017.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[47] Q. Wen, Z. Zhang, Y. Li, and L. Sun, "Fast robuststl: Efficient and robust seasonal-trend decomposition for time series with complex patterns," in *Proc. of KDD*, 2020.

[48] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Poczos, "Mmd gan: Towards deeper understanding of moment matching network," in *Proc. of NeurIPS*, 2017.

[49] Q. Xu, G. Huang, Y. Yuan, C. Guo, Y. Sun, F. Wu, and K. Weinberger, "An empirical study on evaluation metrics of generative adversarial networks," *arXiv preprint arXiv:1806.07755*, 2018.

[50] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019.